representing data

outline

1. encoding data for transmission or storage (bits)
2. encryption and compression
3. compound data: objects and records
4. storing data in a file system: csv files
5. the unix filesystem
6. storing data in a relational database
7. selecting data using relations (SQL query)

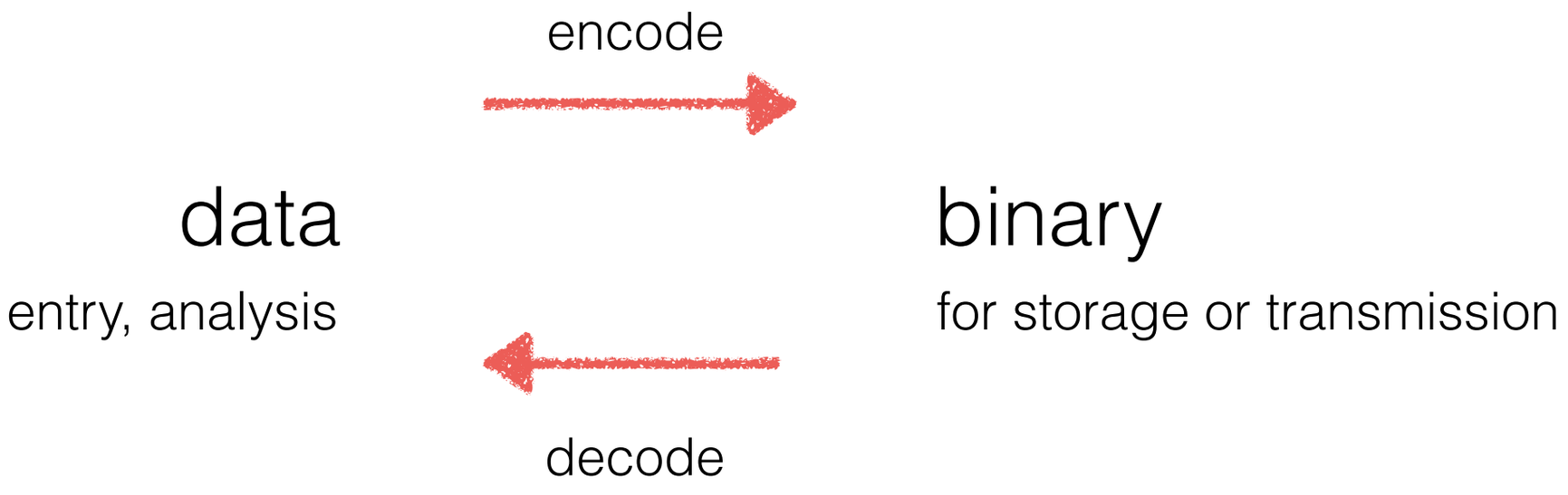
outline

1. encoding data for transmission or storage (bits)
2. encryption and compression
3. compound data: objects and records
4. storing data in a file system: csv files
5. the unix filesystem
6. storing data in a relational database
7. selecting data using relations (SQL query)

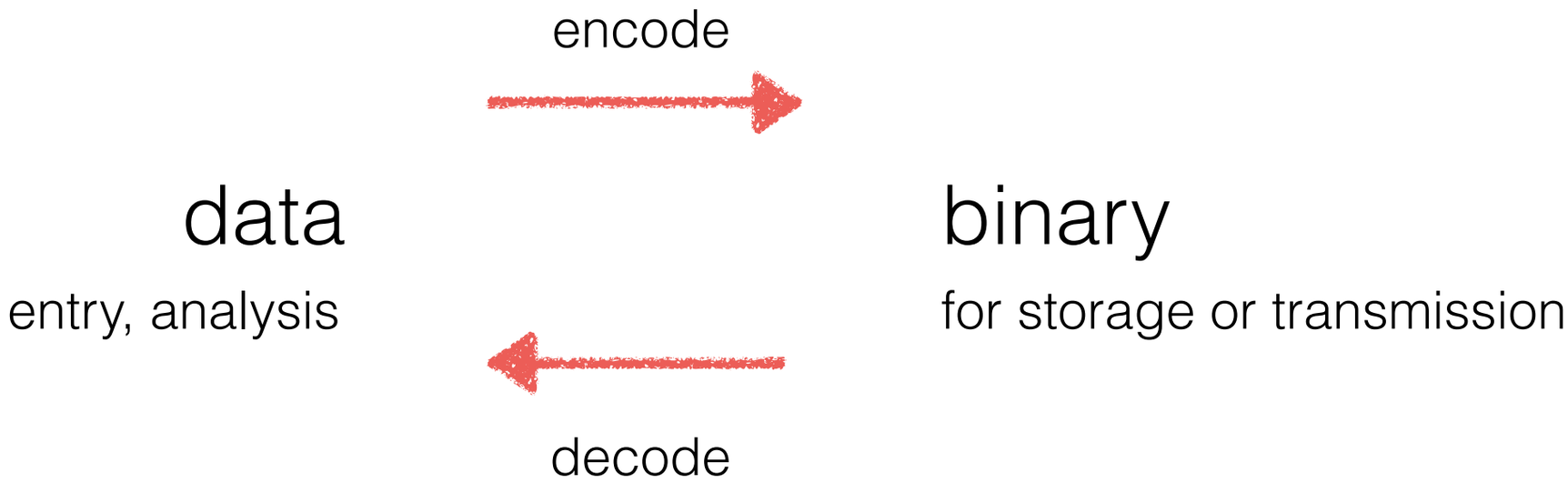
How memory works

01011100 10110101 11011011 00001111 10101010

bit: A binary digit, with the value 0 or 1.
byte: A group of 8 bits
address: the index of the byte
Random Access Memory: constant time access



binary is a **uniform representation** of data
(humans use phonemes)



each data type needs its own not-so-secret code

```
01 01 11 00 10 11 01 01 11 01 10 11
 b  b  d  a  c  d  b  b  d  b  c  d
```

Is this a good code for
representing string values?

code	value
00	a
01	b
10	c
11	d

each data type needs its own **code book**

string
ASCII (8 bits per character) or
unicode (8 to 42 bits per character)

integer
positive: binary counting
negative: two's complement, 64 bits

ASCII Alphabet			
A	1000001	N	1001110
B	1000010	O	1001111
C	1000011	P	1010000
D	1000100	Q	1010001
E	1000101	R	1010010
F	1000110	S	1010011
G	1000111	T	1010100
H	1001000	U	1010101
I	1001001	V	1010110
J	1001010	W	1010111
K	1001011	X	1011000
L	1001100	Y	1011001
M	1001101	Z	1011010

Part of the ASCII alphabet

(from computerhistory.org)

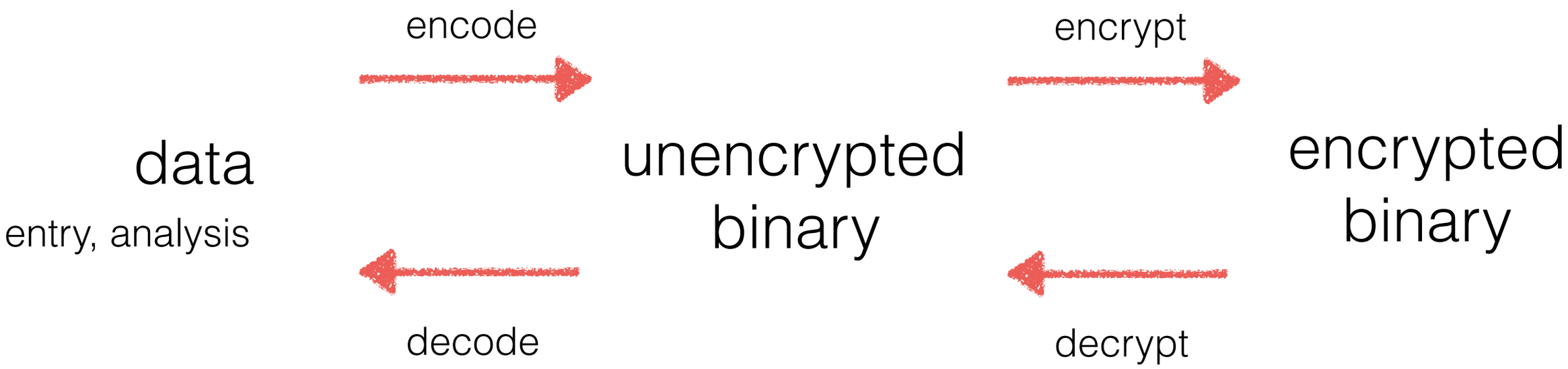
floating point (large or small, exponent)
IEEE floating point standard 754
standard for javascript, but inaccurate due to rounding

fixed point (high-speed traders use this)
Q (Q17.15 has 17 bits for integer, 15 bits for fraction)

outline

1. encoding data for transmission or storage (bits)
2. encryption and compression
3. compound data: objects and records
4. storing data in a file system: csv files
5. the unix filesystem
6. storing data in a relational database
7. selecting data using relations (SQL query)

encryption: **symmetric** vs **asymmetric**



Simple **symmetric** encryption: a secret code book (pad)

Asymmetric encryption (RSA). *Choose two numbers.*

1. you give me your public number (public key)
2. I encrypt: math + your public number (public key)
3. you decrypt: math + your private number (private key)

amazingly, public key allows encryption but won't decrypt

outline

1. encoding data for transmission or storage (bits)
2. encryption and compression
3. compound data: objects and records
4. storing data in a file system: csv files
5. the unix filesystem
6. storing data in a relational database
7. selecting data using relations (SQL query)

compression: zip, jpg, png, mp3, mpeg4

loss-less (zip, png)

frequency: why use 8 bits to represent e and q?

Shannon: how small can you make a file (re-zip?)

lossy (jpg, mp3, mpeg4)

compress some other nicely-compressible data

lose (or add?) high-frequency information

ASCII Alphabet			
A	1000001	N	1001110
B	1000010	O	1001111
C	1000011	P	1010000
D	1000100	Q	1010001
E	1000101	R	1010010
F	1000110	S	1010011
G	1000111	T	1010100
H	1001000	U	1010101
I	1001001	V	1010110
J	1001010	W	1010111
K	1001011	X	1011000
L	1001100	Y	1011001
M	1001101	Z	1011010

Part of the ASCII alphabet

delta (mpeg4)

analyze repetition (between frames)

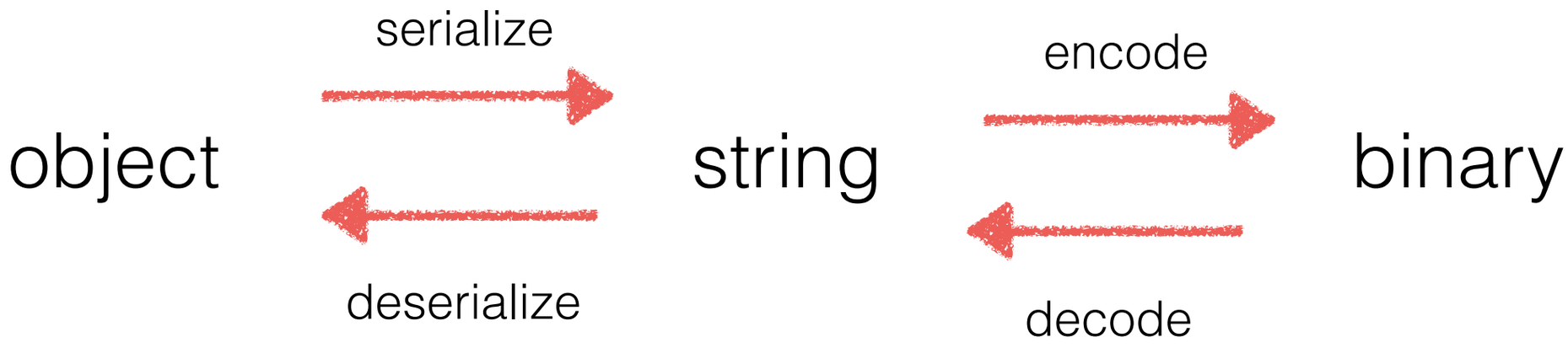
Computer science is not about coding.

What is information? How much do you have? How do you keep it secret? How do you verify or error-check it?

Why is Hany Farid so excited about Benford's law?

outline

1. encoding data for transmission or storage (bits)
2. encryption and compression
3. compound data: objects and records
4. storing data in a file system: csv files
5. the unix filesystem
6. storing data in a relational database
7. selecting data using relations (SQL query)




```
1  // create new empty objects for theDonald and hillary
2  var theDonald = {};
3  var hillary = {};
4
5  theDonald.party = "Republican";
6  theDonald.hairColor = "orange";
7  theDonald.age = 69;
8
9  hillary.party = "Democrat";
10 hillary.hairColor = "blonde";
11 hillary.age = 68;
12
13 print( JSON.stringify(theDonald) );
```

```
{"party":"Republican","hairColor":"orange","age":69}
```

JSON is a set of rules for serialization.
So is XML. (So is HTML.)
So is csv.

outline

1. encoding data for transmission or storage (bits)
2. encryption and compression
3. compound data: objects and records
4. storing data in a file system: csv files
5. the unix filesystem
6. storing data in a relational database
7. selecting data using relations (SQL query)

storing data with .csv files

world.csv

```
1   Afghanistan,AFG,20038215159,31627506
2   Albania,ALB,13211513726,2894475
3   Algeria,DZA,213518000000,38934334
4   Arab World,ARB,2845790000000,385272539
5   Argentina,ARG,537660000000,42980026
6   Armenia,ARM,11644438423,3006154
7   Australia,AUS,1454680000000,23490736
8   Austria,AUT,436888000000,8534492
9   Azerbaijan,AZE,75198010965,9537823
10  Bahrain,BHR,33851063830,1361930
11  Bangladesh,BGD,172887000000,159077513
12  Belarus,BLR,76139250365,9470000
13  Belgium,BEL,531547000000,11225207
14  Bolivia,BOL,32996187988,10561887
15  Bosnia and Herzegovina,BIH,18286273233,3817554
16  Botswana,BWA,15813364345,2219937
17  Brazil,BRA,2346080000000,206077898
```

A1 | fx Afghanistan

	A	B	C	D	E
1	Afghanistan	AFG	2.0038E+10	31627506	
2	Albania	ALB	1.3212E+10	2894475	
3	Algeria	DZA	2.1352E+11	38934334	
4	Arab World	ARB	2.8458E+12	385272539	
5	Argentina	ARG	5.3766E+11	42980026	
6	Armenia	ARM	1.1644E+10	3006154	
7	Australia	AUS	1.4547E+12	23490736	
8	Austria	AUT	4.3689E+11	8534492	
9	Azerbaijan	AZE	7.5198E+10	9537823	
10	Bahrain	BHR	3.3851E+10	1361930	
11	Bangladesh	BGD	1.7289E+11	159077513	
12	Belarus	BLR	7.6139E+10	9470000	
13	Belgium	BEL	5.3155E+11	11225207	
14	Bolivia	BOL	3.2996E+10	10561887	
15	Bosnia and H	BIH	1.8286E+10	3817554	
16	Botswana	BWA	1.5813E+10	2219937	
17	Brazil	BRA	2.3461E+12	206077898	

What's missing from this representation?

A **schema** tells you the structure of the data.
"string: country, string: country abbreviation, number: GDP, number: population"

outline

1. encoding data for transmission or storage (bits)
2. encryption and compression
3. compound data: objects and records
4. storing data in a file system: csv files
5. the unix filesystem
6. storing data in a relational database
7. selecting data using relations (SQL query)

unix terminal and filesystem

Grab data-examples.zip from top of lecture 4 notes and upload to main directory on c9.io. (No need to unzip yet.)

Now go to bash and type the command ls to list files

The unix command unzip <filename> unzips a file.

```
devinbalkcom:~/workspace $ unzip data-examples.zip
Archive:  data-examples.zip
   creating: data-examples/
  inflating: data-examples/countdown.php
  inflating: data-examples/countdown.py
  inflating: data-examples/db
  inflating: data-examples/myusers
  inflating: data-examples/README-SQL.txt
  inflating: data-examples/submit.php
  inflating: data-examples/world.csv
  inflating: data-examples/world.sqlite3
```

```
devinbalkcom:~/workspace $ ls
README.md  data-examples/  data-examples.zip  hello-world.php  php.ini  provided-01.zip  test/
```

unix terminal and filesystem

cd <directoryname> changes the working directory.

```
devinbalkcom:~/workspace $ cd data-examples
devinbalkcom:~/workspace/data-examples $ ls
README-SQL.txt  countdown.php  countdown.py  db  myusers  submit.php  world.csv  world.sqlite3
devinbalkcom:~/workspace/data-examples $
```

side note. Why all the typing? Why are so many nerds using mac or linux? What is linux/unix anyway?

Ok, I'll be nice, use the mouse and c9.io interface to look at the contents of countdown.php.

unix terminal and filesystem

php <filename> runs php code.

```
devinbalkcom:~/workspace/data-examples $ php countdown.php
10
9
8
7
6
5
4
3
2
1
Blast off!
devinbalkcom:~/workspace/data-examples $
```

python <filename> runs python code.

```
devinbalkcom:~/workspace/data-examples $ python countdown.py
```

unix terminal and filesystem

command	what it does
`pwd`	print current directory
`ls`	list files in current directory
`cd dirname`	change to directory
`cd ..`	change to directory above this one
`mkdir dirname`	make a new directory
`touch filename`	make an empty directory
`mv filename1 filename2`	rename/ move a file
`cp filename1 filename2`	copy a file
`rm filename`	remove a file. (-r for directories)

Exercise: create a new directory **tempdir**, and a file **myfile.txt** within.

outline

1. encoding data for transmission or storage (bits)
2. encryption and compression
3. compound data: objects and records
4. storing data in a file system: csv files
5. the unix filesystem
6. storing data in a relational database
7. selecting data using relations (SQL query)

databases

firstName	lastName	email	phone
Devin	Balkcom	devin.balkcom@dartmouth.edu	6-0272
Hany	Farid	hany.farid@dartmouth.edu	6-2761

A **table** is a collection of **rows**. Each row can be thought of as a record: a collection of related information.

Each **column** contains a field: data of a particular type.

relational database query: A search for values over one or more columns returns a set of rows.

creating a database with sqlite3

To create a database, sqlite3 <filename>.
Create a new database db.sqlite3 in data-examples now.

```
devinbalkcom:~/workspace/data-examples $ pwd
/home/ubuntu/workspace/data-examples
devinbalkcom:~/workspace/data-examples $ sqlite3 db.sqlite3
SQLite version 3.8.2 2013-12-06 14:53:30
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
sqlite>
```

Type:

```
CREATE TABLE professors (firstName text,
 lastName text, email text, phone text);
```

creating a database with sqlite3

```
CREATE TABLE professors (firstName text,
 lastName text, email text, phone text);
```

- Notice that a file was just created: db.sqlite3
- You have created a table professors with four columns
- Each column is of type text
- Commands in sqlite end with a semicolon
- By convention, commands are capitalized.

Type `.tables` to list tables. You should have one: professors.

```
Enter SQL statements terminated with a ";"
sqlite> .tables
sqlite> CREATE TABLE professors (firstName text, lastName text, email text, phone text);
sqlite> .tables
professors
sqlite>
```

creating a database with sqlite3

Add a row with `INSERT INTO <table_name> VALUES()`

```
INSERT INTO professors VALUES ('Devin',
'Balkcom', 'devin.balkcom@dartmouth.edu',
'6-0272');
```

Verify that it worked:

```
SELECT * FROM professors;
```

```
sqlite> SELECT * FROM professors
   ...> ;
Devin|Balkcom|devin.balkcom@dartmouth.edu|6-0272
sqlite>
```

Add Hany's info now:
Hany|Farid|hany.farid@dartmouth.edu|6-2761

.commands in sqlite3

command	what it does
`.help`	Lists the .commands
`.quit`	c ya!
`.tables`	list tables
`.schema tablename`	list column names, types for table
`.open filename`	opens the database in filename

Notice, no save command. Saves immediately!

sqlite3 data types

Each value in an SQLite3 database has one of the following *storage classes*.

- NULL. The value has no value
- INTEGER. The value is a signed integer
- REAL. The value is a floating point value
- TEXT. The value is a text string
- BLOB. The value is a blob of data, stored exactly as input (image or music data is not text).

```
CREATE TABLE professors (firstName text,
 lastName text, email text, phone text);
```

sqlite3 primary key

The **primary key** is a column that uniquely identifies each row.

Last name is not a good primary key. SSN is, but private.

```
CREATE TABLE professors2 (id integer PRIMARY KEY
AUTOINCREMENT, firstName text, lastName text, email text,
phone text);
```

To insert:

```
INSERT INTO professors2 VALUES (NULL, 'Devin',
'Balkcom', 'devin.balkcom@dartmouth.edu', '6-0272')
```

Add Hany to professors2 and select all.

a bigger example: world.sqlite3

```
sqlite> .open world.sqlite3
sqlite> .tables
world
sqlite>
```

(To see how world.sqlite3 was created, go to README-SQL.txt)

a bigger example: world.sqlite3

> *CAIN: I'm ready for the 'gotcha' questions and they're already starting to come. And* ***when they ask me who is the president of Ubeki-beki-beki-beki-stan-stan*** *I'm going to say, you know,* ***I don't know. Do you know?***

Let's see what we've got:

```
sqlite> .schema
CREATE TABLE world (country text, abbrv text, gdp real, population real);
sqlite>
```

To get the whole table:

```
SELECT * FROM world;
```

To select a few columns

```
SELECT country, gdp FROM world;
```

```
Uzbekistan|62643953022.0
Vanuatu|814954307.0
Vietnam|186205000000.0
West Bank and Gaza|12737613125.0
Zambia|27066230009.0
Zimbabwe|14196912535.0
sqlite>
```

selecting rows with WHERE

```
SELECT column FROM table WHERE condition;
```

Example:

```
SELECT population FROM world
 WHERE country = FRANCE;
```

Multiple columns and rows:

```
SELECT country, gdp FROM world
 WHERE population < 500000;
```

conditions in SQL

```
SELECT column FROM table WHERE condition;
```

We can use BETWEEN x AND y to narrow further:

```
SELECT country, gdp FROM world WHERE
  population BETWEEN 500000 AND 1000000;
```

Logical operators AND and OR work:

```
SELECT country FROM world WHERE
  population < 500000 AND gdp > 10000000;
```

Exercise: Tuvalu

Write an SQLite command that searches the table world for all countries with a gdp between 10 and 100 million. Your search should return the matching countries' 3-letter abbreviation.

```
SELECT abbrv FROM world WHERE gdp BETWEEN
  10000000 AND 100000000;
```

Nesting searches

Extract names of countries with population larger than France's:

First way: find the population of France with search (66206930), and then use that number (not nested):

```
SELECT country FROM world WHERE population >
66206930;
```

Second way: nest the searches

```
SELECT country FROM world WHERE population >
(SELECT population FROM world WHERE
country='France');
```

Aggregate (calculations on search results)

```
SELECT <command> FROM <table>;
```

Command is usually of the form FUNCTION(column).

```
SELECT SUM(population) FROM world;

SELECT MAX(gdp) FROM world;

SELECT COUNT(*) FROM world;
```

Exercise: wealthy countries

Write a nested search that lists the names and gdp's of all countries with above-average gdp. (Hint -- try something simple first, like computing average gdp.)

```
SELECT country,gdp FROM world WHERE gdp >
(SELECT AVG(gdp) FROM world);
```

```
sqlite> SELECT country,gdp FROM world WHERE gdp > (SELECT AVG(gdp) FROM world);
Arab World|2845790000000.0
Australia|1454680000000.0
Brazil|2346080000000.0
Canada|1785390000000.0
Central Europe and the Baltics|1457320000000.0
Euro area|13410200000000.0
European Union|18514200000000.0
France|2829190000000.0
Germany|3868290000000.0
Indonesia|888538000000.0
Italy|2141160000000.0
Japan|4601460000000.0
Korea|1410380000000.0
Mexico|1294690000000.0
Netherlands|879319000000.0
North America|19210100000000.0
Russian Federation|1860600000000.0
Spain|1381340000000.0
United Kingdom|2988890000000.0
United States|17419000000000.0
sqlite>
```